Secure Android

DASSD, Sept 2022 | C-DAC, Hyderabad

TABLE OF CONTENTS

Let's Begin!

01
Major Components

Android Application Components

- In Android, application components are the basic building blocks of an application and these components will act as an entry point to allow system or user to access our app.

- The **manifest file** contains the app's metadata, its hardware configuration, and platform requirements, external libraries, and required permissions.

Android Application Components

- The following are the basic core application components that can be used in Android application.

Android Application Components

- The following are the basic core application components that can be used in Android application.
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers
  - Intents (for Inter Component Signaling)

Android Application Components

- All these application components are defined in the Android app description file (**AndroidManifest.xml**) like as shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest …..>
    <application android:allowBackup="true" android:icon="@mipmap/ic_launcher" ……>
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
       …….
    </application>
</manifest>
```

Android Activities

- In Android, an **Activity** represents a single screen with a user interface (UI) and it will acts an entry point for the user's to interact with app.

- For example, a contacts app that is having multiple activities like showing a list of contacts, add a new contact, and another activity to search for the contacts. All these activities in the contact app are independent of each other but will work together to provide a better user experience.

Android Activities

- In Android, an **Activity** represents a single screen with a user interface (UI) and it will acts an entry point for the user's to interact with app.

```
public class MainActivity extends Activity {
  // rest of the code
}
```

Android Activities

- The component that most Android developers learn about first is the Activity.

- An Activity is analogous to a single screen displayed on the device to a user that is composed of what the user actually sees. When building an app, you create classes that derive from the Activity base class and typically contain one or more Views, which are objects that users can interact with in some way, such aa reading text displayed within the View, clicking a button, and so on.

- In proper application development terminology, the set of Activities is your application's presentation layer.

Android Services

- In Android, a **Service** is a component that keeps an app running in the background to perform long-running operations based on our requirements.

- For Service, we don't have any user interface and it will run the apps in background like play music in background when the user in different app.

```
public class ServiceName extends Service {
  //rest of the code
}
```

Android Services

- A Service is an Android component that is designed for background processing. These components can run when your app is not visible (that is, none of its Activities are being displayed to the user and are not active). Services typically either carry out some computations or update the data sources for your app.

- For example, if you were to build an email client app, a Service would run in the background to open up connections to the configured mail servers every so often, check for new mail, download it, and store new messages in the local database. Services can also be used to update Activities directly, to provide updates to the user directly via a Notification, or really for any other type of task that needs to occur in the background.

Android Broadcast Receivers

- In Android, Broadcast Receiver is a component that will allow a system to deliver events to the app like sending a low battery message to the app. The apps can also initiate broadcasts to let other apps know that required data available in a device to use it.

- Generally, we use Intents* to deliver broadcast events to other apps and Broadcast Receivers use status bar notifications to let the user know that broadcast event occurs.

```
public class MyReceiver  extends  BroadcastReceiver {
   @override
   public void onReceive(context,intent){}
}
```

Android Broadcast Receivers

- A Broadcast Receiver is a type of component that listens for system messages called Intents. An Intent can be thought of as a request for a certain action to take place. Apps can create Intents and either send them directly to a specific component (usually an Activity or a Service) or broadcast them system-wide to all apps that are running.

- A Broadcast Receiver is a component that can receive these systemwide broadcasts and act upon them; it can choose to listen for all broadcast Intents or set up filters so that it receives only Intents for the specific actions it cares about (and would, presumably, take action upon). As with most broadcast systems, more than one Broadcast Receiver can receive, and act upon, a single Intent.

Android Content Providers

- In Android, Content Providers are useful to exchange the data between the apps based on the requests. The Content Providers can share the app data that stores in the file system, SQLite database, on the web or any other storage location that our app can access.

- By using Content Providers, other apps can query or modify the data of our app based on the permissions provided by content provider. For example, Android provides a Content Provider (ContactsContract.Data) to manage contacts information, by using proper permissions any app can query the content provider to perform read and write operations on contacts information.

Android Content Providers

- A Content Provider is a component designed to share data across apps. You can think of a Content Provider as the public interface to your databases, allowing other apps to connect and either run queries (retrieve data) or issue updates (store data). A Content Provider typically is used as a front end for a database created using the Android standard SQLite database system. As Content Providers are typically used to share data across apps, properly securing them so that appropriate apps can access specific data is **critical**.

```
public class contentProviderName extends  ContentProvider {
   @override
   public void onCreate(){}
}
```

Android Intents

- Intents are the primary way for apps to send messages to other apps (and can also be used to send messages to other components within the same app). To send a message, an app creates an Intent object that signifies that it wants a certain action to take place. This Intent object can specify a specific Activity or Service that you want to start up, or it can specify a specific piece of data. In the later case, the Intent can either specify which components should perform an action on that piece of data, or just ask that someone should. This means that Android allows for Intents that have specific recipients (the Activity or Service that it wishes to interact with) as well as Intents that are broadcast throughout the system to any components that may be listening.

02

UI

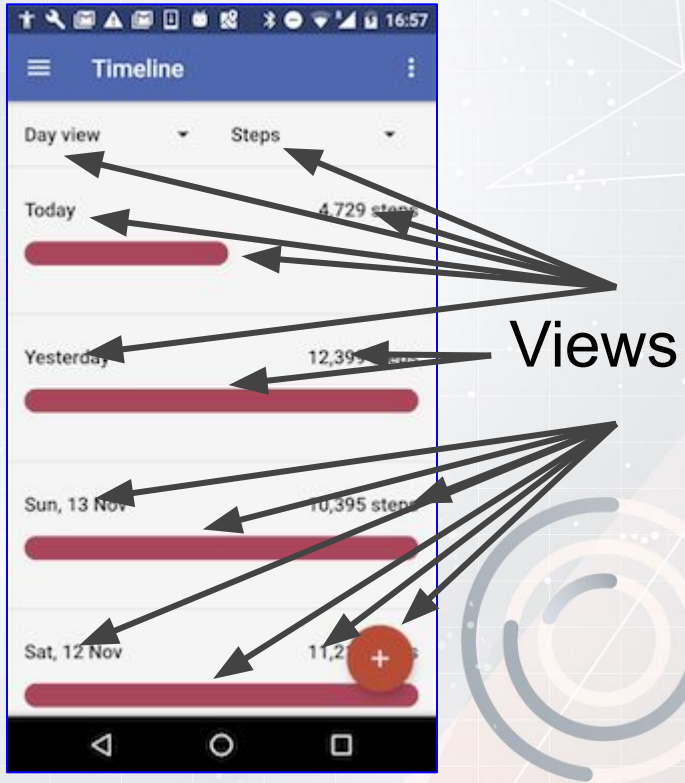
Widgets(Views)

Android Widgets

- Android UI widgets are those components of Android that are used to design the UI in a more interactive way.
- It helps us to develop an application that makes user interaction better with the view components.
- Android provides us a huge range of UI widgets of many types such as buttons, text views, etc.

Everything you see is a view

If you look at your mobile device, every user interface element that you see is a **View**.



What is a view

Views are Android's basic user interface building blocks.

- display text (TextView class), edit text (EditText class)
- buttons (Button class), menus, other controls
- scrollable (ScrollView, RecyclerView)
- show images (ImageView)
- subclass of View class

Views have properties

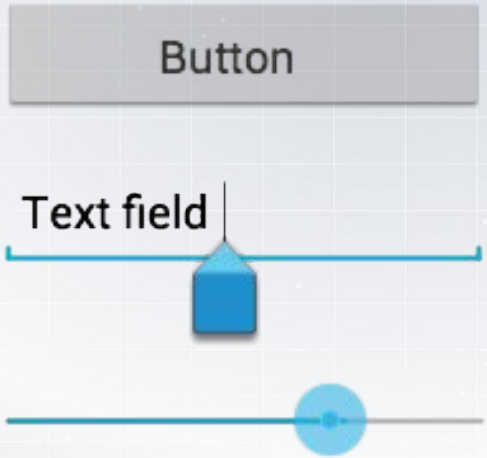
- Have properties (e.g., color, dimensions, positioning)
- May have focus (e.g., selected to receive user input)
- May be interactive (respond to user clicks)
- May be visible or not
- Have relationships to other views

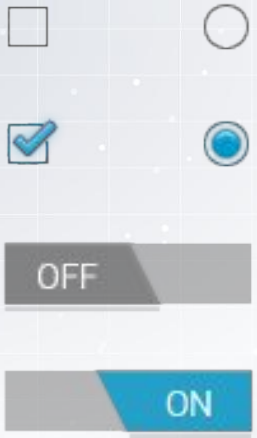
Examples of views

Button

EditText

SeekBar





CheckBox

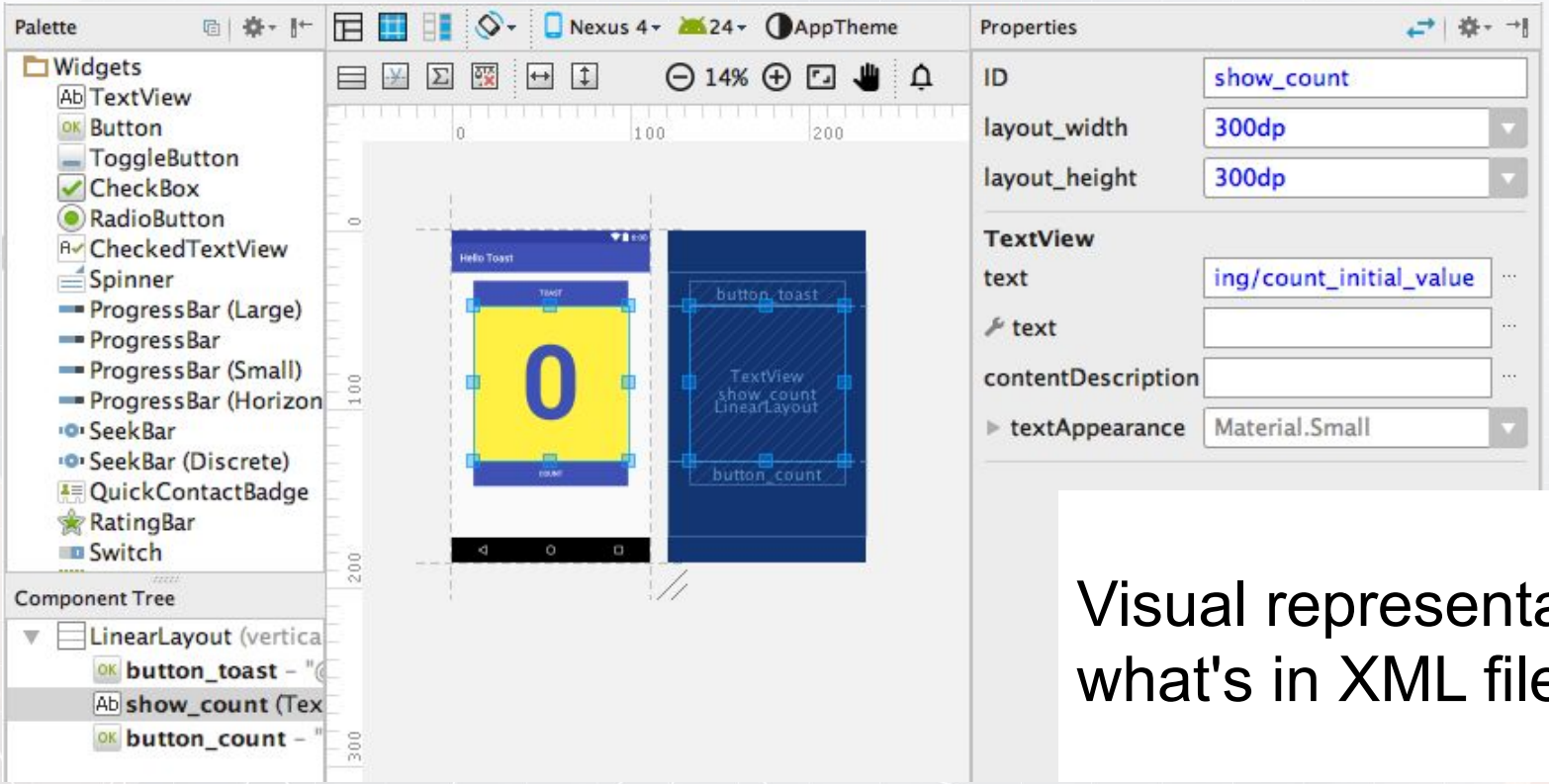
RadioButton

Switch

Creating and laying out views

- Graphically within Android Studio
- XML Files
- Programmatically

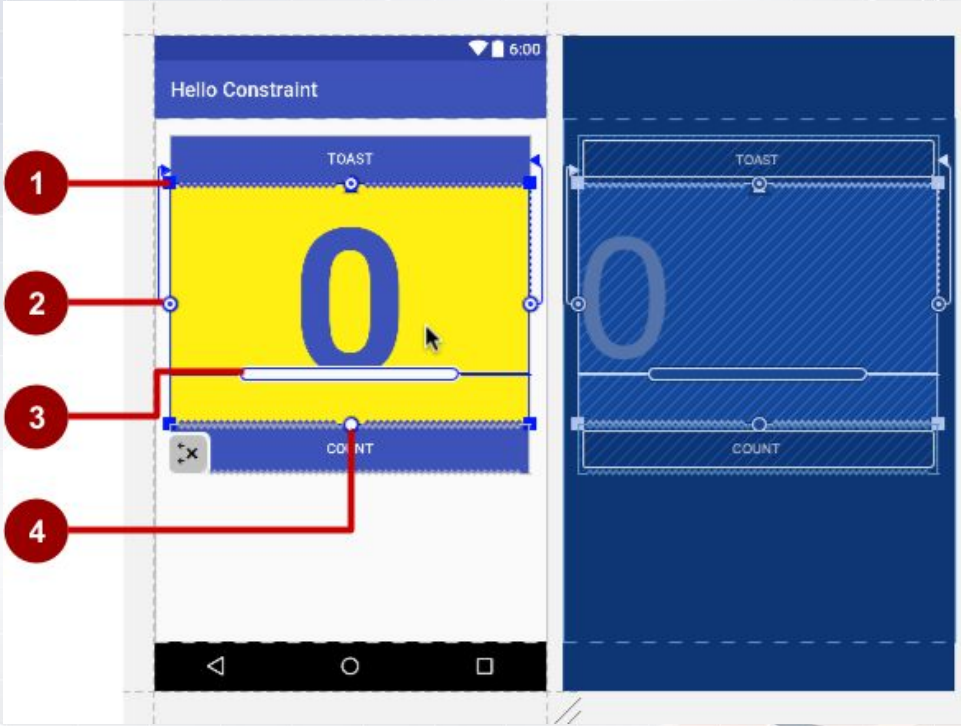
Views defined in Layout Editor



Visual representation of what's in XML file.

Using the Layout Editor

1. Resizing handle
2. Constraint line and handle
3. Baseline handle
4. Constraint handle



Views defined in XML

```
<TextView
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/myBackgroundColor"
        android:text="@string/count_initial_value"
        android:textColor="@color/colorPrimary"
        android:textSize="@dimen/count_text_size"
        android:textStyle="bold"
/>
```

View properties in XML

android:<property_name>="<property_value>"

Example: android:layout_width="match_parent"

android:<property_name>="@<resource_type>/resource_id"

Example: android:text="@string/button_label_next"

android:<property_name>="@+id/view_id"

Example: android:id="@+id/show_count"

View properties in XML

android:<property_name>="<property_value>"

Example: android:layout_width="match_parent"

android:<property_name>="@<resource_type>/resource_id"

Example: android:text="@string/button_label_next"

android:<property_name>="@+id/view_id"

Example: android:id="@+id/show_count"

Create View in Java code

In an Activity:

context

```
TextView myText = new TextView(this);

myText.setText("Display this text!");
```

What is the context?

- Context is an interface to global information about an application environment
- Get the context:

```
Context context = getApplicationContext();
```

- An activity is its own context:

```
TextView myText = new TextView(this);
```

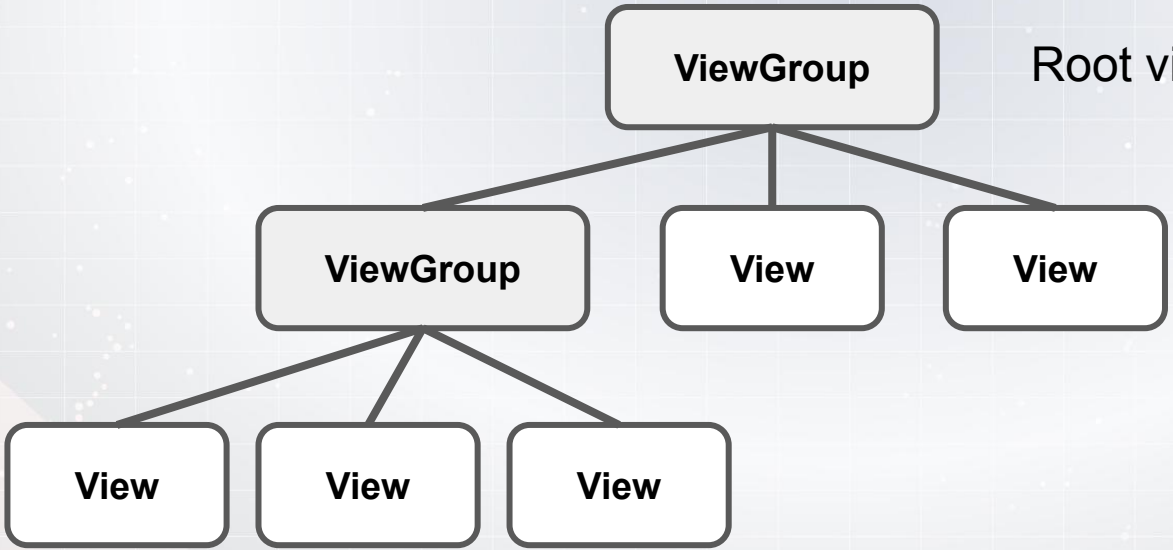
ViewGroup views

A ViewGroup (parent) is a type of view that can contain other views (children)

ViewGroup is the base class for layouts and view containers

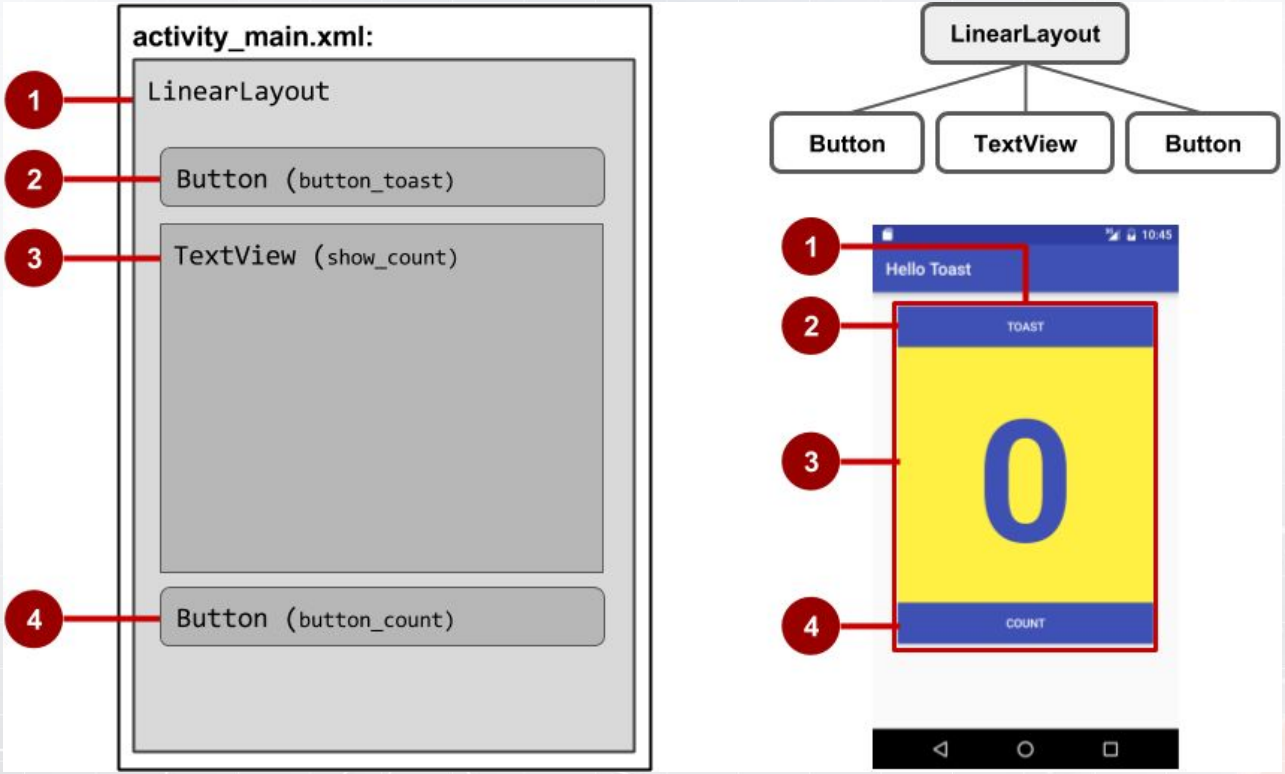
- ScrollView—scrollable view that contains one child view
- LinearLayout—arrange views in horizontal/vertical row
- RecyclerView—scrollable "list" of views or view groups

Hierarchy of view groups and views

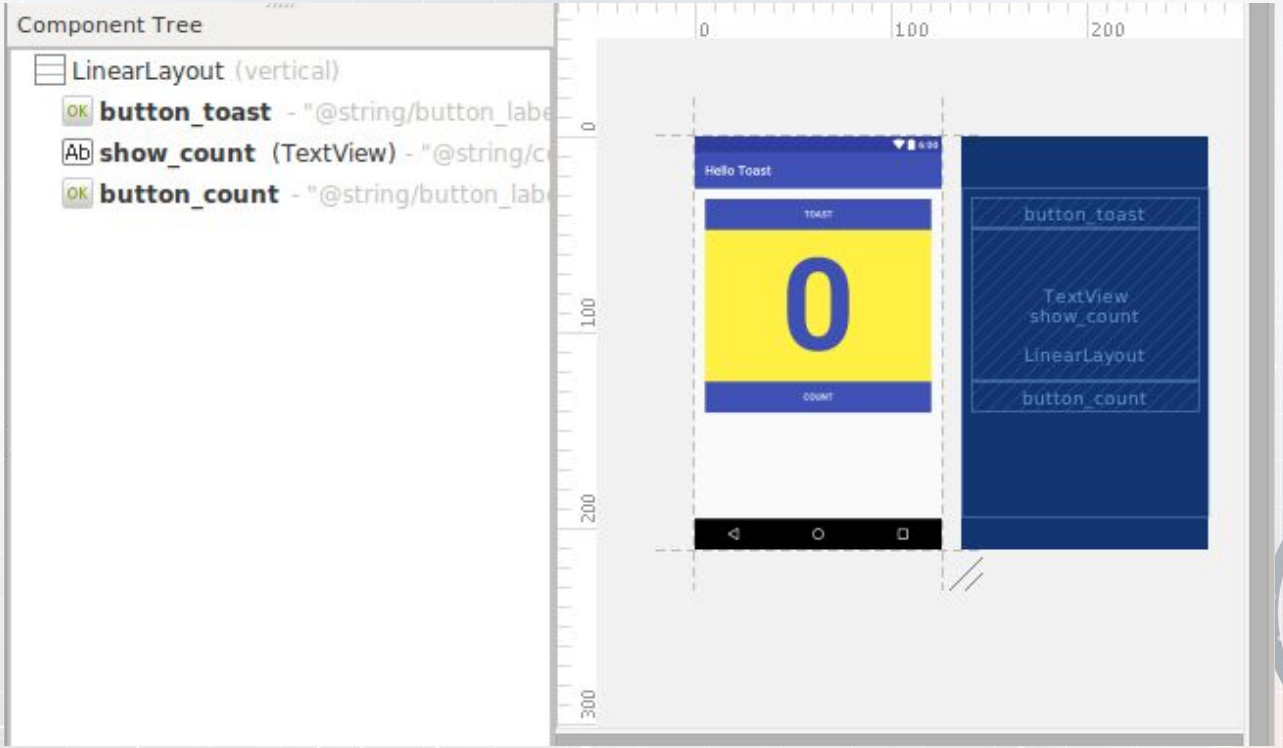


Root view is always a view group

View hierarchy and screen layout



View hierarchy in the component tree



Android Toast

- Android Toast can be used to display information for the short period of time. A toast contains message to be displayed quickly and disappears after sometime.

- In Android, TextView is a user interface control that is used to set and display the text to the user based on our requirements. The TextView control will act as like label control and it won’t allow users to edit the text.

Android Toast

- Android Toast can be used to display information for the short period of time. A toast contains message to be displayed quickly and disappears after sometime.

- In Android, TextView is a user interface control that is used to set and display the text to the user based on our requirements. The TextView control will act as like label control and it won't allow users to edit the text.

```
Toast.makeText(getApplicationContext(),"Hii !! This is a toast message",Toast.LENGTH_SHORT).show();
```

Android TextView

- A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing.

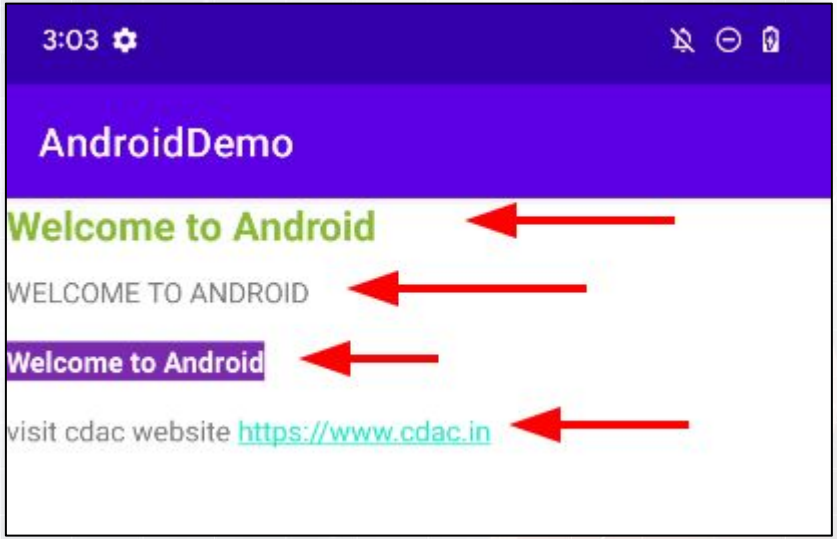
- In Android, TextView is a user interface control that is used to set and display the text to the user based on our requirements. The TextView control will act as like label control and it won’t allow users to edit the text.

Android TextView

```
<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"
    android:text="Welcome to Android"
    android:textStyle="bold"
    android:textColor="#fff"
    android:background="#7F3AB5"

android:layout_marginBottom="15dp"/>
```

```
<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"
    android:autoLink="email|web"
    android:text="visit cdac website
https://www.cdac.in" />
```

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"

android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:text="Welcome to Android"
    android:textColor="#86AD33"
    android:textSize="20dp"
    android:textStyle="bold" />
```



Android TextView

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"
    android:layout_marginBottom="15dp"
    android:textAllCaps="true" />
```


```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView textView = (TextView) findViewById(R.id.textView2);
        textView.setText("Welcome to Android");

    }
}
```

Android TextView Attributes

- The following are some of the commonly used attributes related to TextView control in android applications.

Attribute	Description
android: id	It is used to uniquely identify the control.
android:hint	It is used to display the hint text when text is empty.
android:width	It makes the TextView be exactly this many pixels wide.
android:height	It makes the TextView be exactly this many pixels tall.
android:text	It is used to display the text.
android:textColor	It is used to change the color of the text.

Android TextView Attributes

- The following are some of the commonly used attributes related to TextView control in android applications.

Attribute	Description
android:gravity	It is used to specify how to align the text by the view's x and y-axis.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:textAllCaps	It is used to present the text in all CAPS
android:fontFamily	It is used to specify the font family for the text.

Android EditText

- In Android, EditText is a user interface control which is used to allow the user to enter or modify the text.
- While using EditText control in our android applications, we need to specify the type of data the text field can accept using the **inputType** attribute.

Android EditText

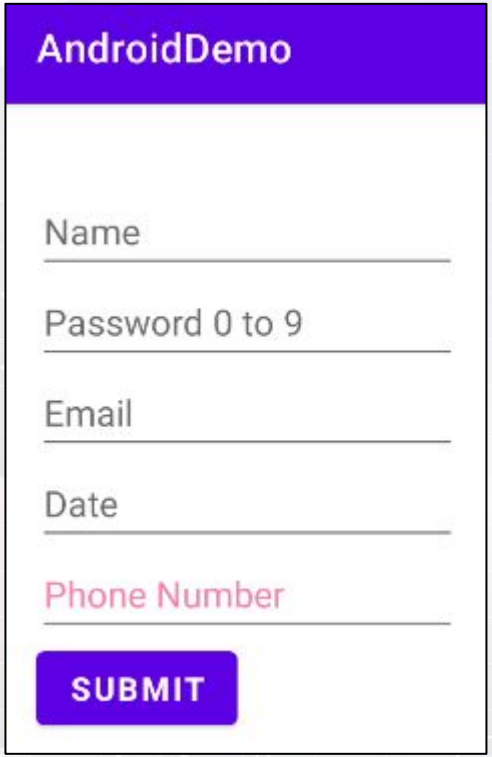
```
<EditText
    android:id="@+id/txtName"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"
    android:layout_marginTop="25dp"
    android:ems="10"
    android:hint="Name"
    android:inputType="text"
    android:selectAllOnFocus="true" />
```

```
<EditText
    android:id="@+id/txtPwd"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Password 0 to 9"
    android:inputType="numberPassword"
/>
```

```
<EditText
    android:id="@+id/txtEmail"
    android:layout_width="wrap_content"

android:layout height="wrap_content"
    android:ems="10"
    android:hint="Email"

android:inputType="textEmailAddress"
/>
```

```
<EditText
    android:id="@+id/txtDate"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout below="@+id/editText3"
    android:ems="10"
    android:hint="Date"
    android:inputType="date" />
```

Android EditText




```
<EditText
    android:id="@+id/txtPhone"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Phone Number"
    android:inputType="phone"
    android:textColorHint="#FE8DAB"/>
```

```
<Button
    android:id="@+id/btnSend"
    android:layout_width="wrap_content"

android:layout_height="wrap_content"
    android:text="submit"
    android:textSize="16sp"
    android:textStyle="normal|bold" />
```

Upon running the app

Android EditText

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText etPhone = (EditText) findViewById(R.id.txtPhone);
        String phone = etPhone.getText().toString();

    }
}
```

Android EditText Attributes

- The following are some of the commonly used attributes related to EditText control in android applications.

Attribute	Description
android: id	It is used to uniquely identify the control.
android:hint	It is used to display the hint text when text is empty.
android:width	It makes the EditText be exactly this many pixels wide.
android:height	It makes the EditText be exactly this many pixels tall.
android:text	It is used to display the text.
android:textColor	It is used to change the color of the text.

Android EditText Attributes

- The following are some of the commonly used attributes related to EditText control in android applications.

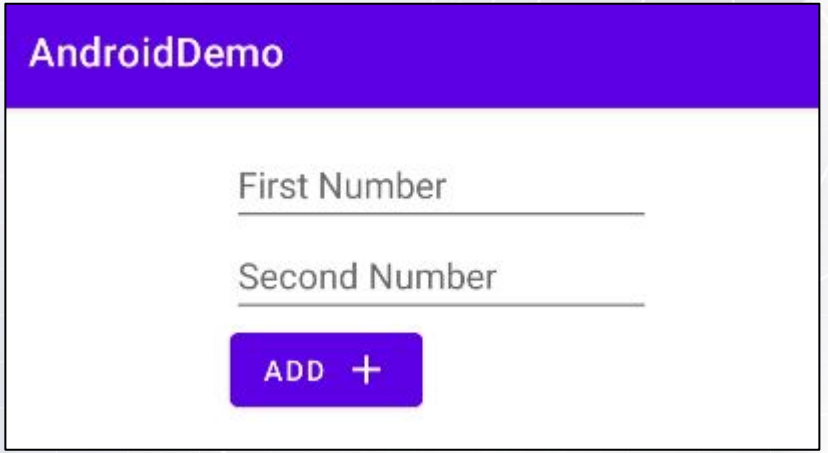
Attribute	Description
android:gravity	It is used to specify how to align the text by the view's x and y-axis.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:textAllCaps	It is used to present the text in all CAPS
android:inputType	It is used to specify the type of text being placed in text fields.

Android Button

- In Android, Button is a user interface control that is used to perform an action whenever the user clicks or tap on it.

- Generally, Buttons in android will contain a text or an icon or both and perform an action when the user touches it.

Android Button

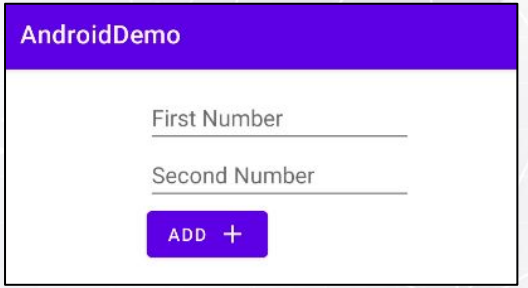



```
<Button
    android:id="@+id/addBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:drawableRight="@drawable/ic_baseline_add_24"
    android:text="Add" />
```

```
<EditText
    android:id="@+id/firstNum"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:hint="First Number"
    android:ems="10" />
```

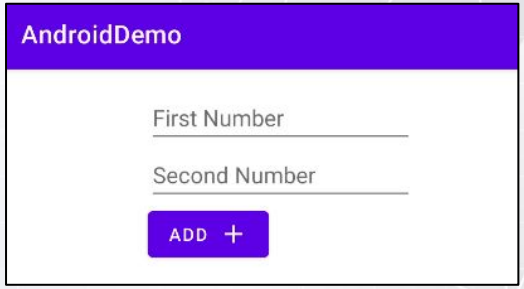
```
<EditText
    android:id="@+id/secondNum"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:hint="Second Number"
    android:ems="10" />
```

Android Button



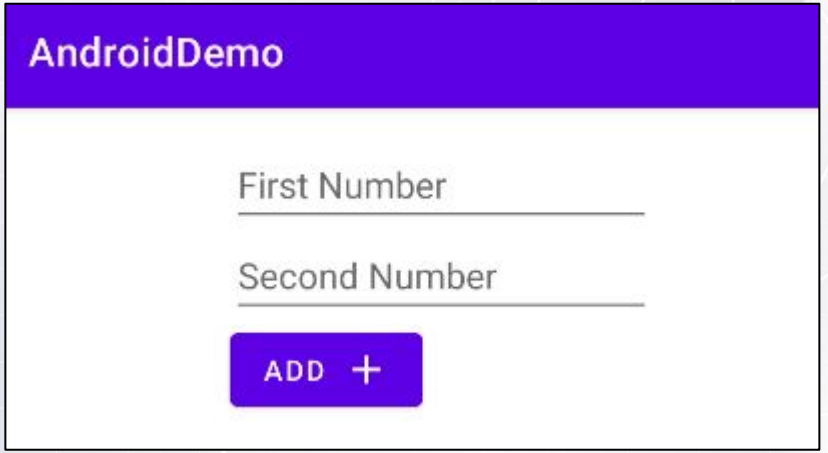

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnAdd = findViewById(R.id.addBtn);

        btnAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //code to be executed upon button click
            }
        });

    }
}
```

Android Button




```
btnAdd.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText etOne = findViewById(R.id.firstNum);
        EditText etTwo = findViewById(R.id.secondNum);

        int firstValue = Integer.parseInt(etOne.getText().toString().toString());
        int secondValue = Integer.parseInt(etTwo.getText().toString().toString());

        int sum = firstValue + secondValue;

        Toast.makeText(MainActivity.this, "Sum is: " + sum, Toast.LENGTH_SHORT).show();
    }
});
```

Android Button



*Defining button click event in XML layout file.

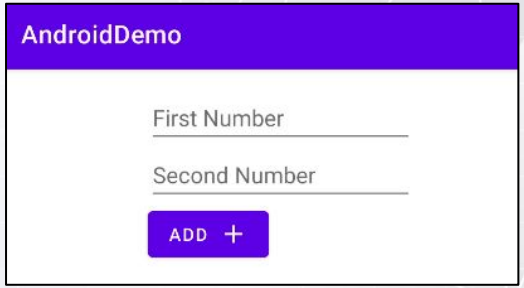
```
<Button
    android:id="@+id/addBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:drawableRight="@drawable/ic_baseline_add_24"
    android:text="Add"
    android:onClick="addOperation" />
```

```
<EditText
    android:id="@+id/firstNum"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:hint="First Number"
    android:ems="10" />
```

```
<EditText
    android:id="@+id/secondNum"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:hint="Second Number"
    android:ems="10" />
```

Android Button

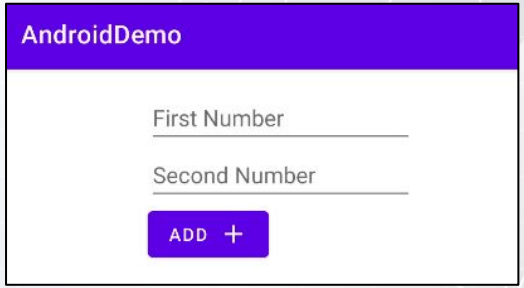
*Defining the corresponding method inside the activity.




```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void addOperation(View view) {
        //code to execute upon button click
    }
}
```

Android Button

*Implementing the functionality inside the method.




```
public void addOperation(View view) {

    EditText etOne = findViewById(R.id.firstNum);
    EditText etTwo = findViewById(R.id.secondNum);

    int firstValue = Integer.parseInt(etOne.getText().toString().toString());
    int secondValue = Integer.parseInt(etTwo.getText().toString().toString());

    int sum = firstValue + secondValue;

    Toast.makeText(MainActivity.this, "Sum is: " + sum, Toast.LENGTH_SHORT).show();

}
```

Android Button Attributes

- The following are some of the commonly used attributes related to Button control in android applications.

Attribute	Description
android: id	It is used to uniquely identify the control.
android:hint	It is used to display the hint text when text is empty.
android:width	It makes the EditText be exactly this many pixels wide.
android:height	It makes the EditText be exactly this many pixels tall.
android:text	It is used to display the text.
android:textColor	It is used to change the color of the text.

Android Button Attributes

- The following are some of the commonly used attributes related to Button control in android applications.

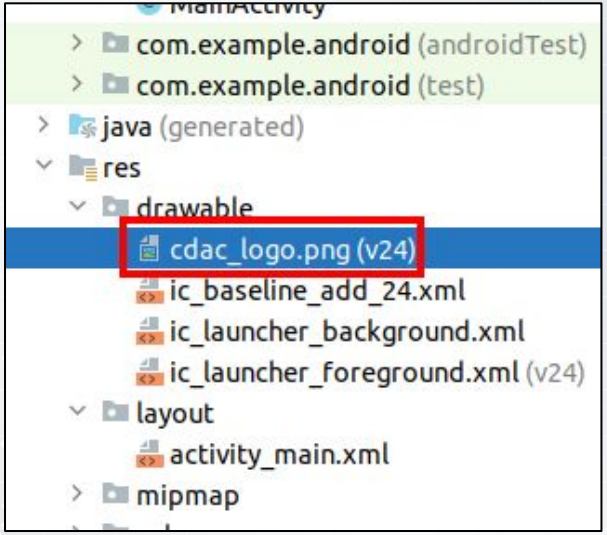
Attribute	Description
android:background	It is used to set the background color for button control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:drawableRight	It's drawable to be drawn to the right of text.
android:drawableLeft	It's drawable to be drawn to the left of the text.
android:onClick	name of the method which we need to call in response to a click event.

Android ImageView

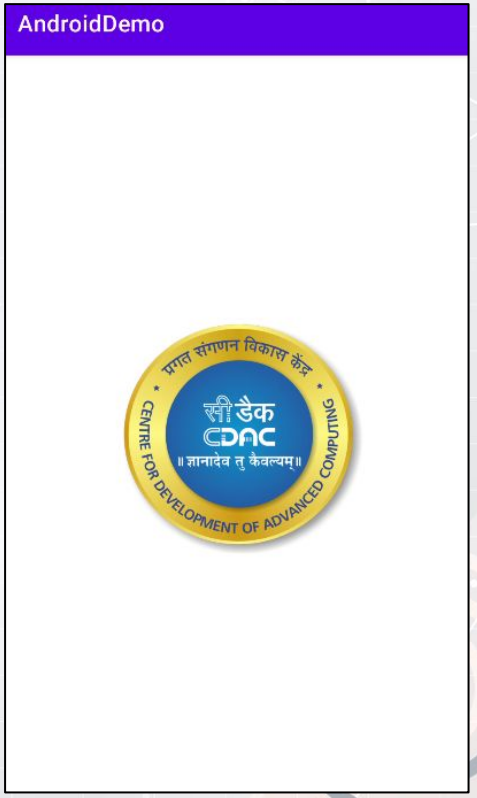
- ImageView is used in Android application to place/show an image in the view.
- Following are some of the main attributes that are most commonly used:

Attribute	Description
android:maxHeight	Used to specify a maximum height for this view.
android:maxWidth	Used to specify a maximum width for this view.
android:src	Sets a drawable as the content for this ImageView.
android:scaleType	Controls how the image should be resized or moved to match the size of the ImageView.
android:tint	Tints the color of the image in the ImageView.

Android ImageView



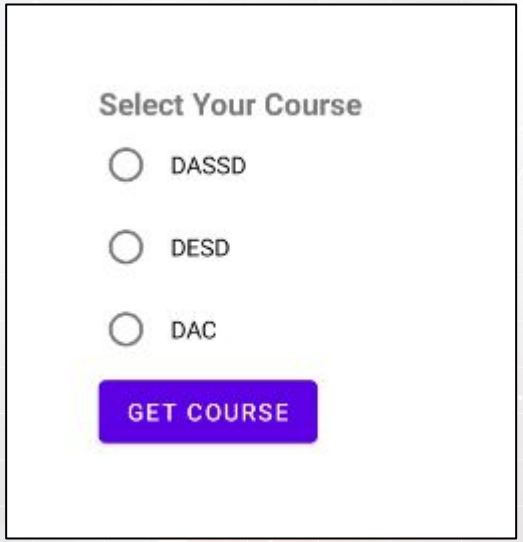

```
<ImageView
    android:id="@+id/img"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scaleType="fitCenter"
    android:src="@drawable/cdac_logo"/>
```



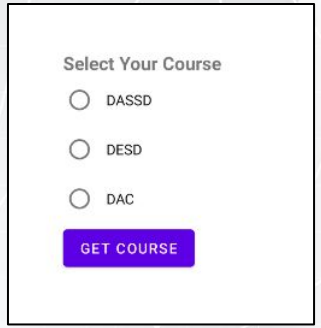
Android Radio Button / Group

- Radio Button is a two-states button that can be either checked or unchecked and allow only one option to select from the group of options at a time.

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
     <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DASSD"
        android:checked="true"/>
```



Android Radio Button / Group



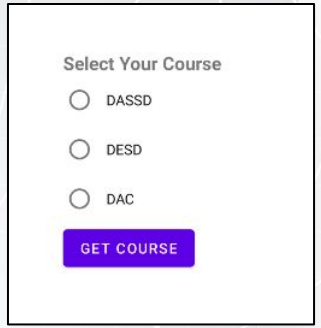

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:id="@+id/rdGroup">


</RadioGroup>
```

```
<RadioButton
    android:id="@+id/rdbDASSD"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginLeft="100dp"
    android:text="DASSD"
    android:onClick="onRadioButtonClicked"/>
```

```
<RadioButton
    android:id="@+id/rdbDAC"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginLeft="100dp"
    android:text="DAC"
    android:onClick="onRadioButtonClicked"/>
```

```
<RadioButton
    android:id="@+id/rdbDESD"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginLeft="100dp"
    android:text="DESD"
    android:onClick="onRadioButtonClicked"/>
```

Android Radio Button / Group

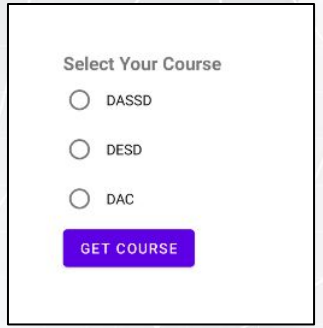


Additionally,

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="150dp"
    android:layout_marginLeft="100dp"
    android:textSize="18dp"
    android:text="Select Your Course"
    android:textStyle="bold"
    android:id="@+id/txtView"/>
```

```
<Button
    android:id="@+id/getBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:text="Get Course" />
```

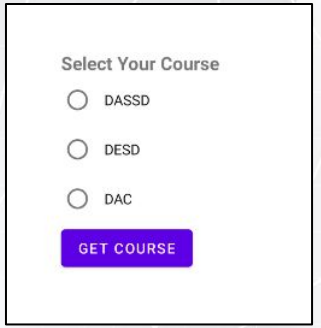
Android Radio Button / Group



On Java side,

```
public class MainActivity extends AppCompatActivity {

    private RadioButton dassd, desd, dac;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        dassd = (RadioButton)findViewById(R.id.rdbDASSD);
        desd = (RadioButton)findViewById(R.id.rdbDESD);
        dac = (RadioButton)findViewById(R.id.rdbDAC);

        Button btn = (Button)findViewById(R.id.getBtn);

    }
```

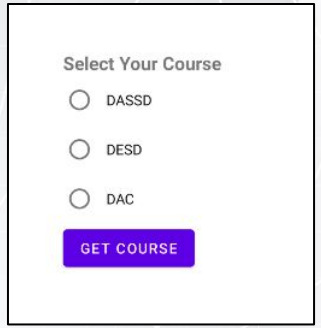
Android Radio Button / Group



On Java side,

```
Button btn = (Button)findViewById(R.id.getBtn);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String result = "Selected Course: ";
        if (dassd.isChecked()){
            result += "DASSD";
        }
        else if (desd.isChecked()){
            result += "DESD";
        }
        else if (dac.isChecked()) {
            result += "DAC";
        }
        Toast.makeText(getApplicationContext(), result, Toast.LENGTH_SHORT).show();
    }
});
```

Android Radio Button / Group



On Java side (additionally),

```
protected void onCreate(Bundle savedInstanceState) {...}

public void onRadioButtonClicked(View view) {
    boolean checked = ((RadioButton) view).isChecked();
    String str="";
    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.rdbDASSD:
            if(checked)
                str = "DASSD Selected";
            break;
        case R.id.rdbDESD:
            if(checked)
                str = "DESD Selected";
            break;
        case R.id.rdbDAC:
            if(checked)
                str = "DAC Selected";
            break;
    }
    Toast.makeText(getApplicationContext(), str, Toast.LENGTH_SHORT).show();
}
```

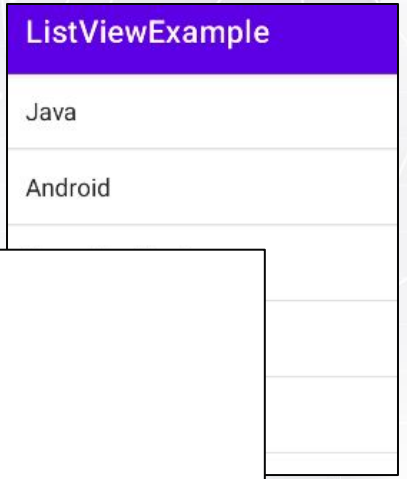
Android List View

- ListView is a view which contains the group of items and displays in a scrollable list.
- ListView uses Adapter classes which add the content from data source (such as string array, array, database etc) to ListView. Adapter bridges data between an AdapterViews and other Views (ListView, ScrollView etc).

Android List View

```
<ListView
    android:id="@+id/my_list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Android List View




```java
public class MainActivity extends AppCompatActivity {

    //1. declare variables
    private ListView myListView;
    private String [] courseModules;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //2. create data source to be displayed in the list view
        courseModules = new String[]{"Java", "Android", "Operating Systems", "Machine Learning", "C Programming"};

        //3. bind the listview layout with the reference variable
        myListView = findViewById(R.id.my_list_view);

        //4. create Array adapter of type string with the following arguments: (context, layout_name, widget_name, array of items to be displayed)
        ArrayAdapter<String> myAdapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, android.R.id.text1, courseModules);

        //5. Finally set the adapter to the list view
        myListView.setAdapter(myAdapter);

    }
}
```

Android List View

Adding onclick() listener to the list items


```
//5. Finally set the adapter to the list view
myListView.setAdapter(myAdapter);

//6. Add click listeners to the list view's items
myListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        //7. Read the value from the position
        String value = adapterView.getItemAtPosition(i).toString();
        //8. Display it or do whatever you want with the value
        Toast.makeText( context: MainActivity.this,  text: "You clicked: " + value, Toast.LENGTH_SHORT).show();
    }
});
```

Android Alert Dialog

- **Android AlertDialog** can be used to display the dialog message with **OK** and **Cancel** buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

- Android AlertDialog is composed of three regions: title, content area and action buttons.
- Methods of AlertDialog class:
  - public AlertDialog.Builder **setTitle**(CharSequence)
  - public AlertDialog.Builder **setMessage**(CharSequence)
  - public AlertDialog.Builder **setIcon**(int)

Android Alert Dialog

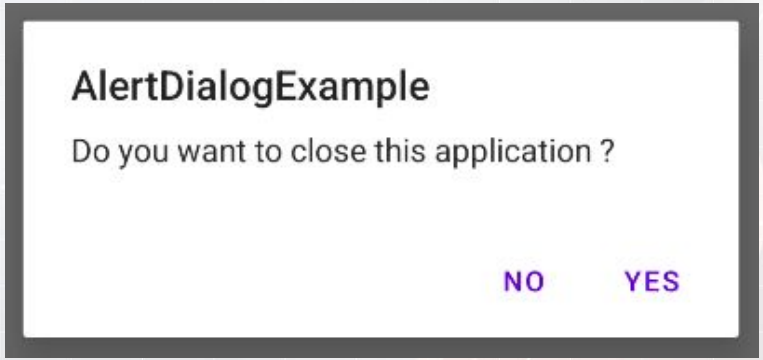
```
<Button
    android:id="@+id/btnShowDialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="show Alert Dialog"/>
```


```
public class MainActivity extends AppCompatActivity {

    //1. create reference variable for the button
    private Button btnShowDialog;
    //2. create a reference variable for the AlertDialog Builder
    AlertDialog.Builder alertDialogBuilder;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //3. Initialize them as follows
        btnShowDialog = findViewById(R.id.btnShowDialog);
        alertDialogBuilder = new AlertDialog.Builder( context: this);

    }
}
```

Android Alert Dialog

```
//4. add click listener on the button to show alert dialog
btnShowDialog.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //5. Write the Alert dialog code here
        alertDialogBuilder.setMessage("Do you want to close this application ?")
                .setCancelable(false) //to prevent the dialog from being dismissed by tapping outside

        1
        .setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                finish(); //to close the app
                Toast.makeText(getApplicationContext(), text: "Yes Button tapped.",
                        Toast.LENGTH_SHORT).show();
            }
        })

        2
        .setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                //  Action for 'NO' Button
                dialog.cancel(); //to dismiss the dialog box
                Toast.makeText(getApplicationContext(), text: "No Button tapped.",
                        Toast.LENGTH_SHORT).show();
            }
        });

        //Creating dialog box
        AlertDialog alert = alertDialogBuilder.create();
        //Setting the title manually
        alert.setTitle("AlertDialogExample");
        alert.show();
    }
});
```

3



03

UI Layouts

Android Layouts

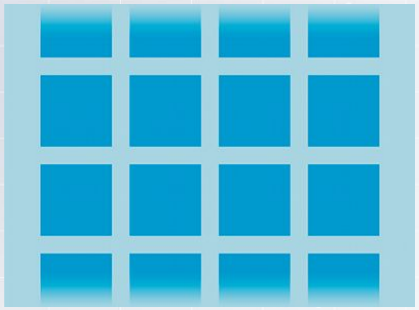
Layouts

- are specific types of view groups
- are subclasses of ViewGroup
- contain child views
- can be in a row, column, grid, table, absolute

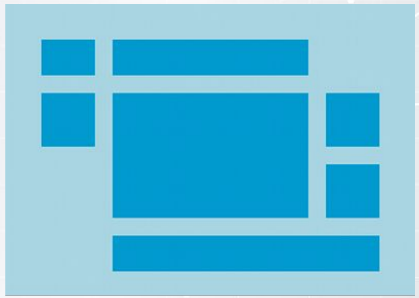
Android Layouts

LinearLayout

RelativeLayout



GridLayout



TableLayout

Common Layout Classes

- **ConstraintLayout -** connect views with constraints
- **LinearLayout** - horizontal or vertical row
- **RelativeLayout** - child views relative to each other
- **TableLayout** - rows and columns
- **FrameLayout** - shows one child of a stack of children
- **GridView** - 2D scrollable grid

Layout created in XML

```
<LinearLayout
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <EditText
        ... />
    <Button
        ... />
</LinearLayout
```

Layout created in Java Activity code

```
LinearLayout linearL = new LinearLayout(this);
linearL.setOrientation(LinearLayout.VERTICAL);

TextView myText = new TextView(this);
myText.setText("Display this text!");

linearL.addView(myText);
setContentView(linearL);
```

Setting width and height in Java code

Set the width and height of a view:

```
LinearLayout.LayoutParams layoutParams =
  new Linear.LayoutParams(
     LayoutParams.MATCH_PARENT,
     LayoutParams.WRAP_CONTENT);

myView.setLayoutParams(layoutParams);
```